IN THE CLAIMS:

The following listing of claims will replace all prior versions, and listings, of claims in the application.

1. (Currently Amended) A method for enabling a user to create a program for controlling a process, wherein the method operates in a system including a computer system which is coupled to the process, the method comprising:

providing a plurality of software classes for modeling, optimization, and deployment;

providing one or more management facilities for managing the plurality of software classes;

creating a plurality of software objects based on the plurality of software classes, wherein said creating the plurality of software objects comprises creating a dynamic process model object and creating a solver object, wherein the dynamic process model object and the solver object comprise a first solution which controls a first process;

creating a model of the first solution;

creating a second solution, wherein the second solution comprises a second dynamic process model object, a second solver object, and the model of the first solution, wherein the second solution which operates to control a second process, wherein the second solution affects the first solution; and

constructing an optimization program which uses the plurality of software objects, wherein said constructing is performed in response to user input;

wherein the optimization program is useable in controlling the process[[.]];

wherein the optimization program is constructed including the dynamic process model object and the solver object, wherein the dynamic process model object and the solver object operate together to optimize the process; and

wherein the second solution uses the model of the first solution to determine how the first solution will respond to actions provided by the second solution.

2. (Original) The method of claim 1, wherein the plurality of software classes include at least one of a modeling class, a simulation class, and an optimization class.

3. (Original) The method of claim 1, wherein the plurality of software classes include at least two of a modeling class, a simulation class, and an optimization class.

4. (Original) The method of claim 3, wherein the plurality of software classes further include two or more of: a visualization class, a deployment class, an execution class, a configuration class, a communication class, a reporting class, a management class, an archiving class, a pre-processing class, and a post-processing class.

5. (Original) The method of claim 1, wherein the plurality of software classes include a modeling class and an optimization class;

wherein said creating the plurality of software objects comprises creating a modeling object and an optimization object;

wherein said constructing the optimization program comprises constructing the optimization program using the modeling object and the optimization object.

6. (Original) The method of claim 1, wherein the plurality of software objects are instances of two or more of the plurality of software classes;

wherein the plurality of software objects comprise primitives that are useable in creating higher level programs.

7. (Original) The method of claim 1, further comprising:

executing the optimization program, wherein said executing comprises executing the plurality of software objects to optimize the process.

8. (Original) The method of claim 7, wherein said executing the optimization program further comprises:

the plurality of software objects communicating with each other using event triggered execution.

9. (Currently Amended) The method of claim 1,

wherein said ~~creating~~ constructing the optimization program comprises creating a decision engine, wherein the decision engine comprises an encapsulation of knowledge and decision making logic;

wherein the decision engine comprises two or more of a modeling object, a simulation object and an optimization object;

wherein the decision engine comprises a modular and portable component.

10. (Original) The method of claim 1, wherein the one or more management facilities comprise one or more of: a global naming facility, a storage and retrieval facility, a cataloging and location facility, a project grouping facility, a deployment facility, a revision tracking facility, and a visualization management facility.

11. (Original) The method of claim 1, wherein the optimization program is operable to perform optimization for two or more of: continuous processes, batch processes and discrete processes.

12. (Original) The method of claim 1, wherein the optimization program is operable to perform control functions for two or more of: continuous processes, batch processes and discrete processes.

13. (Original) The method of claim 1,

wherein said creating a plurality of software objects includes creating a model software object that combines a first-principles model and an empirical model.

14. (Original) The method of claim 1,

wherein said creating a plurality of software objects includes creating a model software object that combines two or more of: a first-principles model, an empirical model, and a procedural model.

15. (Canceled)

16. (Currently Amended) The method of claim [[15]] 1, wherein said creating a plurality of software objects further comprises:

creating a problem formulation object that describes a problem;

wherein the optimization program is constructed including the problem formulation object;

wherein the solver object uses information from the problem formulation object in optimizing the process.

17. (Currently Amended) The method of claim [[15]] 1,

wherein the dynamic process model object comprises one of a first principles model, a linear model, a non-linear model, or a hybrid model;

wherein the solver object comprises one of a nonlinear programming solver, a mixed integer nonlinear programming solver, or an evolutionary solver.

18. (Currently Amended) The method of claim [[15]] 1, wherein said creating a plurality of software objects further comprises:

creating a pre-processing object;

wherein the optimization program is constructed including the pre-processing object;

wherein the pre-processing object is operable to pre-process data prior to provision of the data to the dynamic process model object or the solver object.

19. (Currently Amended) The method of claim [[15]] 1, wherein said creating a plurality of software objects further comprises:

creating a post-processing object;

wherein the optimization program is constructed including the post-processing object;

wherein the post-processing object is operable to post-process data received from one or more of the dynamic process model object or the solver object.

20. (Canceled)

21. (Currently Amended) The method of claim [[20]] 1, further comprising:

the second solution using the model of the first solution to dynamically determine how the first solution will respond to actions provided by the second solution; and

the second solution controlling the first solution based on a determination of how the first solution will respond to actions provided by the second solution.

22. (Currently Amended) The method of claim [[15]] 1, further comprising:

creating a user interface component for the optimization program; and

associating the user interface component with the optimization program.

23. (Original) The method of claim 1, wherein the process comprises one of: an enterprise, a manufacturing operation, or an e-commerce system.

24. (Currently Amended) A control system for controlling a process, wherein the control system comprises:

a computer system including a processor and a memory medium, wherein the computer system is operable to couple to the process;

a plurality of software classes for modeling, optimization, and deployment;

one or more management facilities for managing the plurality of software classes;

a plurality of software objects created in response to the plurality of software classes, wherein the plurality of software objects inherit functionality from one or more of the plurality of software classes, wherein the plurality of software objects comprises a dynamic process model object and a solver object, wherein the dynamic process model object and the solver object comprise a first solution that controls a first process;

a model of the first solution;

a second solution, wherein the second solution comprises a second dynamic process model object, a second solver object, and the model of the first solution, wherein

the second solution operates to control a second process, wherein the second solution affects the first solution; and

a control program created using the plurality of software objects;

wherein the control program is operable to control the process[[.]];

wherein the optimization program comprises the dynamic process model object and the solver object, wherein the dynamic process model object and the solver object operate together to optimize the process; and

wherein the second solution uses the model of the first solution to determine how the first solution will respond to actions provided by the second solution.

25. (Original) The control system of claim 24, wherein the plurality of software classes include at least two of a modeling class, a simulation class, and an optimization class.

26. (Original) The control system of claim 24, wherein the plurality of software classes further include two or more of: a visualization class, a deployment class, an execution class, a configuration class, a communication class, a reporting class, a management class, an archiving class, a pre-processing class, and a post-processing class.

27. (Original) The control system of claim 24, wherein the plurality of software classes include a modeling class and an optimization class;

wherein the plurality of software objects comprise a modeling object and an optimization object;

wherein the control program comprises the modeling object and the optimization object.

28. (Original) The control system of claim 24, wherein, during execution of the control program, the plurality of software objects communicate with each other using event triggered execution.

29. (Original) The control system of claim 24,

wherein the control program comprises a decision engine, wherein the decision engine comprises an encapsulation of knowledge and decision making logic;

wherein the decision engine comprises two or more of a modeling object, a simulation object and an optimization object;

wherein the decision engine comprises a modular and portable component.

30. (Original) The control system of claim 24, wherein the control program is operable to perform control functions for two or more of: continuous processes, batch processes and discrete processes.

31. (Canceled)

32. (Currently Amended) The control system of claim [[31]] 24, wherein the plurality of software objects further comprises a problem formulation object that describes a problem;

wherein the optimization program comprises the problem formulation object;

wherein the solver object uses information from the problem formulation object in optimizing the process.

33. (Currently Amended) The control system of claim [[31]] 24,

wherein the dynamic process model object comprises one of a first principles model, a linear model, a non-linear model, or a hybrid model;

wherein the solver object comprises one of a nonlinear programming solver, a mixed integer nonlinear programming solver, or an evolutionary solver.

34. (Currently Amended) The control system of claim [[31]] 24, wherein the plurality of software objects further comprises a pre-processing object;

wherein the optimization program comprises the pre-processing object;

wherein the pre-processing object is operable to pre-process data prior to provision of the data to the dynamic process model object or the solver object.

35. (Currently Amended) The control system of claim [[31]] 24, wherein the plurality of software objects further comprises a post-processing object;

wherein the optimization program comprises the post-processing object;

wherein the post-processing object is operable to post-process data received from one or more of the dynamic process model object or the solver object.

36. (Canceled)

37. (Currently Amended) The control system of claim [[36]] 24,

wherein the second solution controls the first solution based on a determination of how the first solution will respond to actions provided by the second solution.

38. (Currently Amended) The control system of claim [[31]] 24, further comprising:

a user interface component for the control program;

wherein the user interface component is operable to be associated with the control program.

39. (Original) The control system of claim 24, wherein the process comprises one of: an enterprise, a manufacturing operation, or an e-commerce system.

40. (Original) The control system of claim 24, wherein the control system comprises a plurality of computer systems interconnected via a network.

41. (Canceled)

42. (Canceled)

43. (Currently Amended) A method for enabling a user to create a program for controlling a process, wherein the method operates in a system including a computer system which is coupled to the process, the method comprising:

providing a plurality of software classes for modeling, optimization, and deployment;

providing one or more management facilities for managing the plurality of software classes; and

creating a plurality of software objects based on the plurality of software classes, wherein said creating a plurality of software objects comprises:

creating a dynamic process model object; and

creating a solver object; [[and]]

wherein the dynamic process model object and the solver object comprise a first solution that controls a first process;

the method further comprising:

creating a second solution, wherein the second solution comprises a second dynamic process model object and a second solver object which operate together to control a second process, wherein the first solution affects the second solution;

creating a model of the second solution;

including the model of the second solution into the first solution, wherein the first solution uses the model of the second solution to determine how the second solution will respond to actions provided by the first solution; and

constructing an optimization program which uses the plurality of software objects, wherein said constructing is performed in response to user input, wherein the optimization program is constructed including the dynamic process model object and the solver object, wherein the dynamic process model object and the solver object operate together to optimize the process.

44. (Original) The method of claim 43, wherein said creating a plurality of software objects further comprises:

creating a problem formulation object that describes a problem;

wherein the optimization program is constructed including the problem formulation object;

wherein the solver object uses information from the problem formulation object in optimizing the process.

45. (Original) The method of claim 43,

wherein the dynamic process model object comprises one of a first principles model, a linear model, a non-linear model, or a hybrid model;

wherein the solver object comprises one of a nonlinear programming solver, a mixed integer nonlinear programming solver, or an evolutionary solver.

46. (Original) The method of claim 43, wherein said creating a plurality of software objects further comprises:

creating a pre-processing object;

wherein the optimization program is constructed including the pre-processing object;

wherein the pre-processing object is operable to pre-process data prior to provision of the data to the dynamic process model object or the solver object.

47. (Original) The method of claim 43, wherein said creating a plurality of software objects further comprises:

creating a post-processing object;

wherein the optimization program is constructed including the post-processing object;

wherein the post-processing object is operable to post-process data received from one or more of the dynamic process model object or the solver object.

48. (Canceled)

49. (Original) The method of claim 43, further comprising:

the first solution using the model of the second solution to dynamically determine how the second solution will respond to actions provided by the first solution; and

the first solution controlling the second solution based on a determination of how the second solution will respond to actions provided by the first solution.

50. (Original) The method of claim 43, further comprising:

creating a user interface component for the optimization program; and

associating the user interface component with the optimization program.

51. (Original) The method of claim 43, wherein the process comprises one of: an enterprise, a manufacturing operation, or an e-commerce system.

52. (Currently Amended) A method for modeling a process, wherein the method operates in a system including a computer system, the method comprising:

creating a software program that implements a model, wherein the model combines aspects of a first-principles model and an empirical model, wherein the empirical model is created from a regression of data;

executing the software program to model the process.

53. (Original) The method of claim 52, wherein said executing comprises executing aspects of the first-principles model and the empirical model.

54. (Original) The method of claim 52, further comprising:

training the model, wherein said training uses empirical data and first-principles information.

55. (Original) The method of claim 52, further comprising:

identifying parameters of the first-principles model based on empirical data, wherein said identifying uses nonlinear empirical modeling techniques.

56. (Original) The method of claim 52,

wherein the model combines aspects of a first-principles model, an empirical model, and a procedural model.

57. (Currently Amended) A method for optimizing a process, the method comprising:

receiving user input which configures constraints and targets of the process as a trajectory, wherein the trajectory comprises a profile which is dependent on time;

controlling the process based on the configured constraints and targets of the process, wherein the process is controlled to ~~substantially conform to~~ follow the trajectory.

58. (Original) The method of claim 57,

wherein said controlling the process comprises optimizing the process according to the configured constraints and targets of the process, wherein the process is optimized according to the trajectory.

59. (Original) A method for enabling a user to create a program for controlling a process, wherein the method operates in a system including a computer system which is coupled to the process, the method comprising:

creating a first solution comprising a first dynamic process model and a first solver, wherein the first solution controls a first process;

creating a model of the first solution;

creating a second solution, wherein the second solution comprises a second dynamic process model, a second solver, and the model of the first solution, wherein the second solution controls a second process, wherein the second solution affects the first solution;

wherein the second solution uses the model of the first solution to determine how the first solution will respond to actions provided by the second solution.

60. (Original) The method of claim 59, further comprising:

the second solution using the model of the first solution to dynamically determine how the first solution will respond to actions provided by the second solution; and

the second solution controlling the first solution based on a determination of how the first solution will respond to actions provided by the second solution.

61. (Original) A method for enabling a user to create a program for controlling a process, wherein the method operates in a system including a computer system which is coupled to the process, the method comprising:

creating a first solution comprising a first dynamic process model and a first solver, wherein the first solution controls a first process;

creating a second solution, wherein the second solution comprises a second dynamic process model and a second solver, wherein the second solution controls a second process, wherein the first solution affects the second solution;

creating a model of the second solution;

including the model of the second solution into the first solution;

wherein the first solution uses the model of the second solution to dynamically determine how the second solution will respond to actions provided by the first solution.

62. (Original The method of claim 61, further comprising:

the first solution using the model of the second solution to dynamically determine how the second solution will respond to actions provided by the first solution; and

the first solution controlling the second solution based on a determination of how the second solution will respond to actions provided by the first solution.

63. (New) A memory medium comprising program instructions which are executable by a processor to:

provide a plurality of software classes for modeling, optimization, and deployment;

provide one or more management facilities for managing the plurality of software classes;

create a plurality of software objects based on the plurality of software classes, wherein, in said creating the plurality of software objects, the program instructions are further executable by the processor to create a dynamic process model object and create a solver object, wherein the dynamic process model object and the solver object comprise a first solution that controls a first process;

create a model of the first solution;

create a second solution, wherein the second solution comprises a second dynamic process model object, a second solver object, and the model of the first solution, wherein the second solution which operates to control a second process, wherein the second solution affects the first solution; and

construct an optimization program which uses the plurality of software objects, wherein said constructing is performed in response to user input;

wherein the optimization program is useable in controlling the process;

wherein the optimization program is constructed including the dynamic process model object and the solver object, wherein the dynamic process model object and the solver object operate together to optimize the process; and

wherein the second solution uses the model of the first solution to determine how the first solution will respond to actions provided by the second solution.

64. (New) The memory medium of claim 63, wherein the plurality of software classes include at least one of a modeling class, a simulation class, and an optimization class.

65. (New) The memory medium of claim 63, wherein the plurality of software classes include at least two of a modeling class, a simulation class, and an optimization class.

66. (New) The memory medium of claim 65, wherein the plurality of software classes further include two or more of: a visualization class, a deployment class, an

execution class, a configuration class, a communication class, a reporting class, a management class, an archiving class, a pre-processing class, and a post-processing class.

67. (New) The memory medium of claim 63, wherein the plurality of software classes include a modeling class and an optimization class;

wherein, in said creating the plurality of software objects, the program instructions are further executable by the processor to create a modeling object and an optimization object;

wherein, in said constructing the optimization program, the program instructions are further executable by the processor to construct the optimization program using the modeling object and the optimization object.

68. (New) The memory medium of claim 63, wherein the plurality of software objects are instances of two or more of the plurality of software classes;

wherein the plurality of software objects comprise primitives that are useable in creating higher level programs.

69. (New) The memory medium of claim 63, wherein the program instructions are further executable by the processor to:

execute the optimization program, wherein, in said executing the optimization program, the program instructions are further executable by the processor to execute the plurality of software objects to optimize the process.